Java中的线程池

2021年12月29日 9:25

线程池实现原理

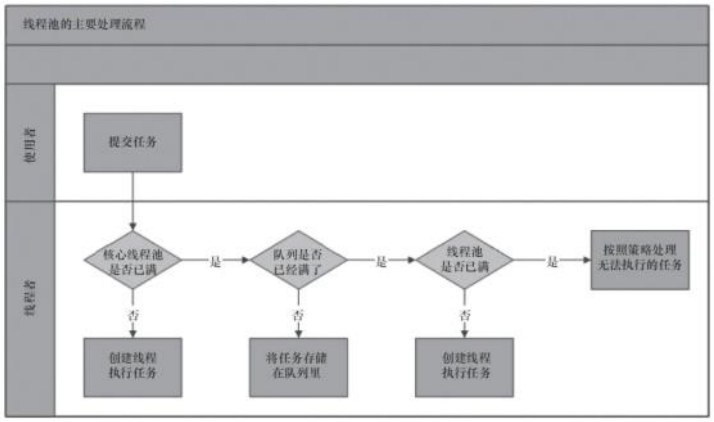

图9-1 线程池的主要处理流程

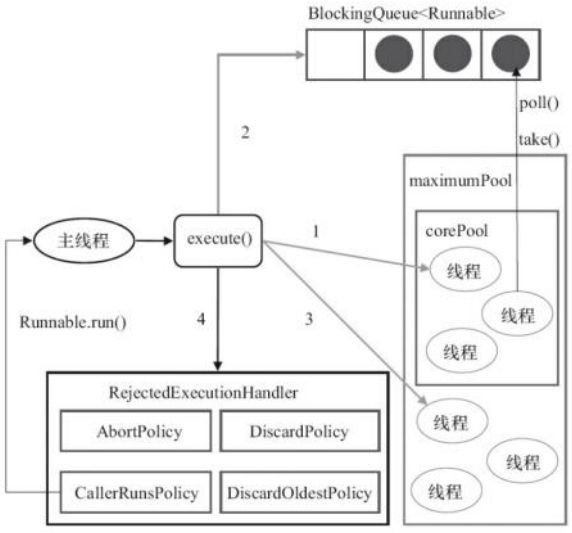

图9-2 ThreadPoolExecutor执行示意图

ThreadPoolExecutor执行execute方法分下面4种情况。

1）如果当前运行的线程少于corePoolSize，则创建新线程来执行任务（注意，执行这一步骤需要获取全局锁）。

2）如果运行的线程等于或多于corePoolSize，则将任务加入BlockingQueue。

3）如果无法将任务加入BlockingQueue（队列已满），则创建新的线程来处理任务（注意，执行这一步骤需要获取全局锁）。

4）如果创建新线程将使当前运行的线程超出maximumPoolSize，任务将被拒绝，并调用RejectedExecutionHandler.rejectedExecution()方法。

- AbortPolicy：直接抛出异常。
- CallerRunsPolicy：只用调用者所在线程来运行任务。
- DiscardOldestPolicy：丢弃队列里最近的一个任务，并执行当前任务。
- DiscardPolicy：不处理，丢弃掉。

合理配置线程池

①任务性质

① CPU密集. N+1

② I/O密集. 2*N.

③ 混合型 → 若拆分为①和②时间差不多则拆分能提高效率.

②优先级.

采用 PriorityBlockingQueue.

③执行时间不同.

① 采用 PriorityBlockingQueue.
② 交给不同规模的线程池处理.

④任务的依赖性（例：依赖数据库连接）

依赖数据库连接池的任务，因为线程提交SQL后需要等待数据库返回结果，等待的时间越长，则CPU空闲时间就越长，那么线程数应该设置得越大，这样才能更好地利用CPU。

⑤经验.

① 使用有界队列.

稳定性个，预警能力个.

若无界，异常情况会撑爆内存，导致系统不可用.